Attorney Docket No. 10559-401001
Application No. 09/823,235
Amendment dated April 12, 2004
Reply to Office Action dated January 14, 2004

Amendments to the Specification:

Please replace the paragraph beginning at page 2, line 25 with the following amended paragraph:

A1

To increase execution speed, stages of pipeline 15 can operate at the same time on different instructions. For example, while decode stage 152 is decoding an instruction, fetch stage 151 can fetch another instruction from instruction cache 14. However, the decision as to which instruction to fetch is often based on the results of previous instructions. For example, depending on the results of the instructions preceding a branch instruction (e.g., an if-statement), a branch in the sequence of instructions may or may not be taken. Fetch stage 151 may use a branch prediction algorithm to determine whether the next instruction to fetch is sequentially the next instruction in the program sequence. The next instruction in the program sequence is also called a fall-through instruction, which fetched if the branch is predicted not taken. If the predicted branch is ~~predicted~~ taken, an instruction at the branch target is fetched.

Please replace the paragraph beginning at page 3, line 15 with the following amended paragraph:

A2

If branch is mispredicted, the instructions fetched by mistake, between the time when the branch was fetched and when the branch is computed in execute stage 154, need to be removed from the pipeline. Consequently, a long latency will occur for pipeline 15 to remove the partially executed fetched instruction and to retrieve the actual next instruction. This latency is usually called a branch misprediction penalty.

Please replace the paragraph beginning at page 4, line 18 with the following amended paragraph:

A3

Referring to FIG. 1, processor 11 includes a DAG trace cache 22 for storing DAG traces. Each DAG trace contains information about a group of interdependent instructions among which data dependency exists. The information stored in DAG trace 22, as will be described in detail below with reference to FIG. 4, allows pipeline 15 to dynamically predict or pre-compute ~~in~~ an outcome of a criterion instruction in attempt to prevent the criterion instruction from incurring long latency. The interdependent instructions include a criterion instruction, which is either a branch instruction or a load instruction that can incur long latency when executed by pipeline 15. The

A3

interdependent instructions also include the instructions, called associated instructions, from which the criterion instruction has data dependence. For example, assume that a branch bases its outcome on the sign of a value V (e.g., if V>0, then instruction_block_1, else instruction_block_2). Prior to the branch, V is assigned the value of a register R. The criterion instruction, in this example, is the branch. The associated instructions include the instruction that assigns the value of register R to V, and, if any, the instructions that modify the value of register R before it is assigned to V. DAG trace cache 22 can be stored as a separate entity from instruction cache 14, or can be logically embedded as part of an instruction cache 14 and reside in any cache lines of the instruction cache that are marked as part of the trace cache. Processor 11 further includes a trace builder 21 that constructs the DAG traces from the instructions stored in instruction cache 14 or from the committed instructions and their execution results generated by pipeline 15. The traces built by trace builder 21 are placed in trace cache 22.